Evaluating a Qualitative Reasoner

Sam Waugh[1], Tim Menzies[2], and Simon Goss[1]

[1] Air Operations Division, Defence Science and Technology Organisation, Melbourne, Australia; {sam.waugh,simon.goss}@dsto.defence.gov.au
[2] Artificial Intelligence Department, School of Computer Science & Engineering, University of NSW, Sydney, Australia; timm@cse.unsw.edu.au

Abstract. In order to support verification, validation and analysis of dynamic Operations Research (OR) models a method of testing models against data is required. In the case of the QCM qualitative reasoning system (QRS) this requires an extension to accommodate temporal data streams. This paper examines a number of temporal reasoning methods for QCM. On the basis of this a general methodology for evaluating QRSs and a statement of success criteria have been developed, and will be used in future work.

1 Introduction

This paper examines a number of methods for extending an existing qualitative reasoning system (QRS), the QCM algorithm [11], to handle time-based reasoning. This system contains a static model compiler, a data compiler, and a hypothesis tester for the model and data. QCM has been able to detect previously invisible errors in theories published in the international neuroendocrinological literature [4, 11]. However, QCM is restricted to *non-temporal* theories with the invariant that no variable can have two different values. In the case of time-based simulation, this invariant is inappropriate since variables can have different values at different times. One way to extend QCM to time-based simulation is to rename variables at each time point in the simulation; e.g. $population$ could be renamed to $population1, population2 \ldots populationT$ where T is some time point. Once these renamed variables are created, the design issue becomes "how should we best *link* variables at time i to time j?". In this paper, we assess eight possible *linking policies* for TQCM, a QCM variant which allows the processing of time series data.

Overview articles which contrast different approaches to qualitative reasoning (QR) (e.g. [2, 3, 5, 7, 11, 12]) have little to say about how to choose between different systems. There is also a lack of guidelines for developing and testing new modelling approaches. In the process of testing the different TQCM linking policies we have developed a more general framework for assessing QRSs. We identify *critical success metrics* (CSMs) for QR, and develop a test engine to collect the CSMs for the different variants of TQCM. In order for this testing to be comprehensive, a wide range of representative models must be examined. A *model mutator* is used to generate large numbers of these test models. We argue that this QR evaluation framework is a process applicable to many domains. However, (i) it is only practical after automating the test engine and (ii) it is only reliable if the model mutator covers a sufficient range.

The following sections detail the CSMs (§2); QCM (§3) and eight linking policies for implementing TQCM (§3.1); the test engine for calculating performance (§4); and present our results (§5), discussion (§6) and conclusions (§7).

2 Critical Success Metrics

This section develops success criteria for a QRS; i.e. an ideal QRS must be *accurate, restrictive* and *practical*. A QRS translates the continuous variables in quantitative models to a small number of discrete values [5]. In order to test if the translation is valid then (success criteria #1) an *accurate* QRS must be able to reproduce the known behaviour of the quantitative system it is modelling. For the purposes of validation, (success criteria #2) a *restrictive* QRS must be able to exclude a significant percentage of impossible behaviours.

The resulting qualitative model is less defined than the original quantitative model, and combinations of poorly-defined influences may be undefined. For example, consider two continuous variables whose qualitative representation is taken from the sign of the first derivative of their values; i.e. `up, down` or `steady`. If we add two increasing values, the result must also be increasing; i.e. `up + up = up`. However, it is unclear what will result from certain other combinations; e.g. `up + down = up` or `down` or `steady`. This is the "chatter" problem. QRSs generate the superset of behaviours possible from a model [6]. The generation of these extra behaviours takes time and can cripple a QRS. Therefore, (success criteria #3) a *practical* QRS must tame the chatter problem. In practice, chatter reduces restrictiveness by the generation of such behaviours.

We can visualise the satisfaction of these criteria in Fig. 1. Consider an human operator trying to express an understanding of a quantitative model in a qualitative approximation. We say that their qualitative model is `good` if it can explain `everything` observed in the quantitative model (point `A`: success criteria of accuracy); and we say their model is `poor` if it can explain `nothing` (point `B`: success criteria of restrictiveness). As the qualitative model degrades from `good` to `poor` we would like to see `curve1`; i.e. we quickly get feedback that we can explain progressively less and less of the behaviour of the quantitative model. However, even `curve2` would satisfy the success criteria of restrictiveness. Lastly, we would declare the chatter problem to be manageable if qualitative indeterminacy does not cripple the QRS; i.e. we achieve low runtimes (the dotted line) when we run the system on a poor model (success criteria of practicality).

3 QCM: A Qualitative Reasoning System

QCM takes a graph-theoretic view of QR. Qualitative statements from domain experts are treated like macros that contribute edges to a search space. For example, if the expert says "weight gain encourages heart disease and exercise reduces weight gain" then (i) we would record it as `weightGain ++ heartDisease` and `exercise -- weightGain` and (ii) expand it internally into the search space of Fig. 2. This space is then searched by an abductive inference engine looking for consistent connections

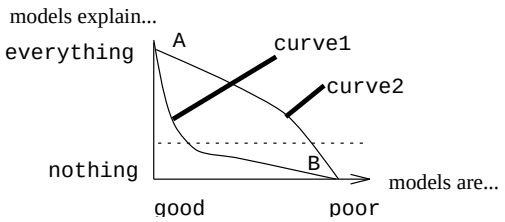

Fig. 1. Visualising success

between known inputs and known outputs. QR indeterminacy is handled by generating different extensions. For full details, see [11].

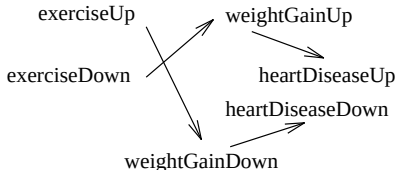

Fig. 2. Weight gain, heart disease, and exercise

QCM is much simpler than other QR approaches (e.g. time averaging [7], consolidation [3], first-order logic qualitative modelling [2], non-linear QR [12], or linear equation QR [5] such as QSIM [6]). Menzies and Compton [10] remark that seemingly naive systems can in fact produce satisfactory competency with far less effort than alternative "more sophisticated" approaches. QCM has detected errors in theories of neuroendocrinology published in the international peer-reviewed journals that were invisible to both the reviewers of those journals and the authors of those theories [4, 11]. Hence we say that is if QCM works in our domain (i.e. satisfies the test criteria of §2), then we lack the motivation to explore more complex approaches.

3.1 Linking Options

This section discusses extensions to QCM to handle time-based simulation. QCM cannot validate theories performing time-based simulations since it assumes that it is inconsistent to believe that objects like heartDisease have multiple values. While this is a valid assumption for non-time based inferencing, a variable in a simulation run can take multiple values over time, e.g. a sensor maintaining a real-world quantity.

In order to handle the temporal simulations, we extend QCM to TQCM. In TQCM one copy of the model is created for each time step in the simulation, as each model represents the processes occurring at one time step. The question then becomes how can

we sensibly link these copies of the search space together? That is, where do we place our links between models? We can identify three types of linking strategies: (1) linking via nodes or via edges; (2) implicit versus explicit time notation; and (3) varying the linking look ahead.

1) Node vs. Edge Linking: In node linking an "appropriate" model node is somehow identified and directly linked to future instances of that node; for example, `X` is expanded to `X(t=i) ++ X(t=i+1)`[1]. Alternatively, we could link by picking an edge in the model consistent with a time connection, then creating a new edge with the starting node of the edge in one time step and the end node the subsequent time step; for example, `X to Y` is expanded to `X(t=i) to Y(t=i+1)`[2]. This method is called edge linking.

2) Implicit vs. Explicit Linking: "Appropriate" nodes or edges can be manually chosen by the model author to explicitly link models (given a "`*`" marking). Alternatively, an implicit linking process can decide that every edge or node is "appropriate". Explicit linking requires some domain knowledge as to where linking is sensible, but results in much fewer links between time steps which reflects more semantic knowledge about the domain.

Implicit node linking ($\texttt{TQCM}^{\texttt{inode}}$) was explored by Menzies & Cohen [8, 9]. In $\texttt{TQCM}^{\texttt{inode}}$, every node at time `i` is connected to the node of the same name at time `i+1`. The regularity of this linking policy permits some general statements about the computational complexity of generating proof trees over time-series simulations. $\texttt{TQCM}^{\texttt{inode}}$ has the interesting property that if a proof cannot be generated in 3 copies, it can never be generated at all. This 3-copy-limit may be used to significantly optimise the search strategy.

3) Future Linking: How far ahead should we place time edges? The obvious solution is to just connect ahead one time step. However, without any domain knowledge stating this to be the case, we need to consider the possibility that a node should be connected to all subsequent steps. That is, for times `i=1 to N` link copy `i` to `f` where `f` is a *future* copy that will take values `i+1...N`. Alternative combinations of forward linking are possible, but will not be considered here.

Table 1 summarises the eight linking options, including abbreviations.

4 A Test Engine

Our test engine has four sub-routines: (1) representative model selection; (2) data generation; (3) model mutation; and (4) option exploration.

1) *Representative model selection*: The "fisheries simulation model" (Fig. 3) is similar to models developed for pursuit and surveillance in the military domain. It includes feedback loops; qualitative states; and measurable entities. A precise mathematical expression of this model is available [1, pp135-141].

2) *Data generation*: The selected qualitative model was run 15 times over five time steps to generate numeric test data using different input parameters to create an array of quantitative observations `measure[1..15]`. From each comparison of `measure[i]`

[1] `X(t=i)` represents a value `X` at time `i`.

[2] Let "`to`" denote an edge annotated as either `++` or `--`.

Table 1. Summary of linking policies. `f` denotes some time from `i+1...N` where `N` is the last time copy.

Linking style	*Abbreviation*	*Theory feature*	*New time links*
implicit edge	$\texttt{TQCM}^{\texttt{iedge}}$	`X to Y`	`X(t=i) to Y(t=i+1)`
implicit edge forward	$\texttt{TQCM}^{\texttt{iedgef}}$	`X to Y`	`X(t=i) to Y(t=f)`
implicit node	$\texttt{TQCM}^{\texttt{inode}}$	`X`	`X(t=i) ++ X(t=i+1)`
implicit node forward	$\texttt{TQCM}^{\texttt{inodef}}$	`X`	`X(t=i) ++ X(t=f)`
explicit edge	$\texttt{TQCM}^{\texttt{xedge}}$	`X to* Y`	`X(t=i) to Y(t=i+1)`
explicit edge forward	$\texttt{TQCM}^{\texttt{xedge}}$	`X to* Y`	`X(t=i) to Y(t=f)`
explicit node	$\texttt{TQCM}^{\texttt{xnode}}$	`X*`	`X(t=i) ++ X(t=i+1)`
explicit node forward	$\texttt{TQCM}^{\texttt{xnodef}}$	`X*`	`X(t=i) ++ X(t=f)`

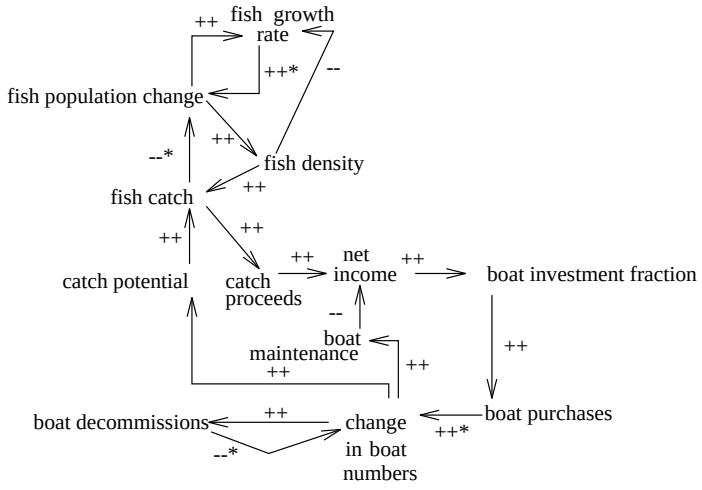

Fig. 3. The fisheries model showing explicit time edges. Adapted from [1, pp135-141].

with measure[j] ($i < j \leq 15$), 105 entries were written to an array of qualitative observations changes[1..105]. For example, if in comparison change[33], the fish density h was increased and the fish catch b was always seen to decrease at all time steps, then change[33].in is h=up and change[33].out is b(t=1)=down, b(t=2)=down, b(t=3)=down, b(t=4)=down, b(t=5)=down.

3) *Model mutation*: This process must be repeated for a large number of representative models from a domain. As these are hard to find in practice, we generate them using a *mutation strategy*, in which a random sample of X statements in the qualitative form of the known representative model are corrupted. Given a model with E edges, then as we vary X from 0 to E, we are moving from a good model to a poor model; i.e. the x-axis of Fig. 1. We corrupted the model by flipping the annotation on an edge (e.g. ++ to -- or visa versa). The corruption-model-mutator picks its edges to corrupt at random, and we repeat the corruption a statistically significant number of times (20 repeats). The only exceptions are when X=0 or E, where there is only a single possible model. Fisheries has 17 edges which allows us to generate 2^{17} possible theories (including the original).

4) *Option exploration*: We created six model copies copy[0..5]. Copy[i] was connected to copy[i+1] (and latter copies for forward linking) according to each of the eight linking policies. Change inputs were mapped into copy[0]. Change outputs were mapped into some copy[1..5]. The success of each run was assessed using the generated data, by recording the percentage of the *explicable outputs* i.e. those outputs that the model could connect back to inputs. Returning to Fig. 1, everything=100% explicable and nothing=0% explicable. Proofs for outputs at time $T = 5$ must be consistent with proofs from $T = 1 \ldots 4$. Hence, all the proofs must be built together (see runQualitativeModel in Fig. 4). For this study, we only collected percentage explicable figures for outputs at time $T = 5$. The final experimental design is shown in Fig. 4.

5 Results

Figure 5 shows the results of applying the test engine to the fisheries model. The success of each linking policy was assessed via comparing its plot against the goal plot of curve1 in Fig. 1.

$\texttt{TQCM}^{\texttt{xnode}}$ was the clear winner showing accuracy and restrictiveness closest to the ideal of curve1. However these linking policies could still offer explanations for about 20% of data, even for very poor models. We attribute these *residual explanations* to the indeterminacy of qualitative models.

$\texttt{TQCM}^{\texttt{iedge}}$ was nearly as accurate as explicit node linking, but could never explain 100% of the behaviour of uncorrupted models (maximum explicable=85%). $\texttt{TQCM}^{\texttt{xedge}}$ proved to be not accurate as, even on good models, it could not explain most outputs. $\texttt{TQCM}^{\texttt{inode}}$ was not sufficiently restrictive as, even on poor models, it could explain most outputs.

Forward linking provided no advantages for the fisheries model. $\texttt{TQCM}^{\texttt{inode}}$ and $\texttt{TQCM}^{\texttt{inodef}}$, and $\texttt{TQCM}^{\texttt{xnode}}$ and $\texttt{TQCM}^{\texttt{xnodef}}$ behaved in virtually the same way. $\texttt{TQCM}^{\texttt{iedgef}}$ was inferior to $\texttt{TQCM}^{\texttt{iedge}}$ since it was far less restrictive than $\texttt{TQCM}^{\texttt{iedge}}$. $\texttt{TQCM}^{\texttt{xedgef}}$ was

```
Inputs:  1) the quantitative fisheries model M0
         2) the qualitative fisheries model M1 with E := 17 edges
         3) T               := 5 maximum time steps
         4) linkingPolicies := [iedge,iedgef,xedge,xedgef,
                                inode,inodef,xnode,xnodef]
Outputs: explicable, runtime

measure[1..15] := runQuantitativeModel(T,M0)
change[1..105] := comparisons(measure)
for policy ∈ linkingPolicies do
  for corrupted := 0 to E do
    if corrupted = 0 or E then repeats := 1 else repeats := 20
    for r := 1 to repeats do
      M2 := corruptSomeEdgesChoosenAtRandom(corrupted,M1)
      for t := 0 to T   do copy[t]:= M2 done
      for t := 0 to T-1 do timeConnect(copy[t],copy[t+1],policy) done
      for i := 1 to |change| do
        <in,out[1..T]>  := change[i]
        startTime       := timeNow()
        explained[1..T] := runQualitativeModel(copy,in,out)
        runtime[policy,r,corrupted,i]    := timeNow() - startTime
        explicable[policy,r,corrupted,i] := |explained[T]|*100/|out[T]|
done done done done
```

Fig. 4. Experimental design for fisheries model

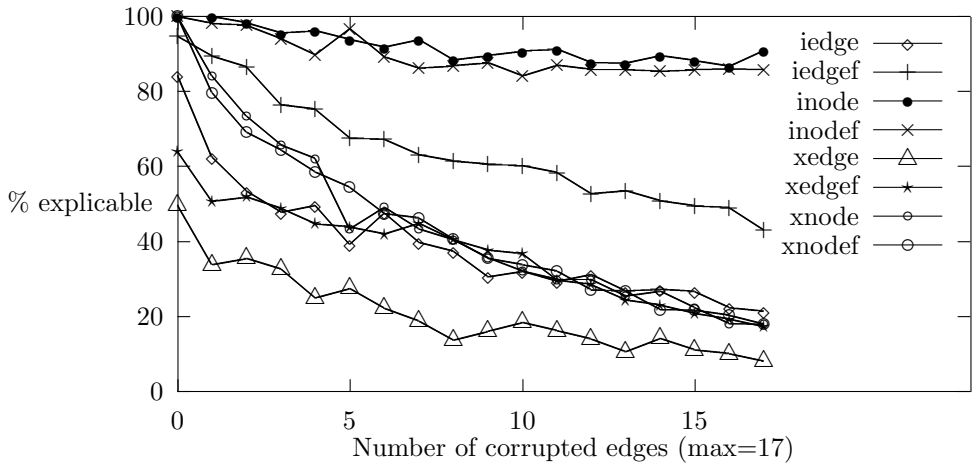

Fig. 5. CSM results for fisheries model

more permissive than TQCM$^{\texttt{xedge}}$, but its maximum explicable rate was too low to be useful (62%). We speculate that forward linking was not useful was due to the nature of the fisheries model. The equations of fisheries all assumed a "one-year-lookahead" as its time step increments. We speculate that models with time links of variable delays may benefit more from forward linking.

For the satisfactory linking policies (TQCM$^{\texttt{xnode}}$, TQCM$^{\texttt{iedge}}$), after only a third of the model being mutated, only around half the outputs are inexplicable. This is a nice result: we get clear, early indications if we are straying from a good model.

The average runtimes in seconds for each trial are shown in Fig. 6. Forward linking was always slower than non-forward linking. The satisfactory linking policies had similar runtimes. However, TQCM$^{\texttt{xnode}}$ and TQCM$^{\texttt{xedge}}$ were fastest since these define the smallest number of time links and, hence, the smallest search space to explore. Hence, we are satisfied that the success criteria of practicality is satisfied. One interesting feature of Fig. 6 is that as the models grow more corrupted, it becomes faster to determine what outputs are explicable. This is a another nice result: we can reject nonsense faster.

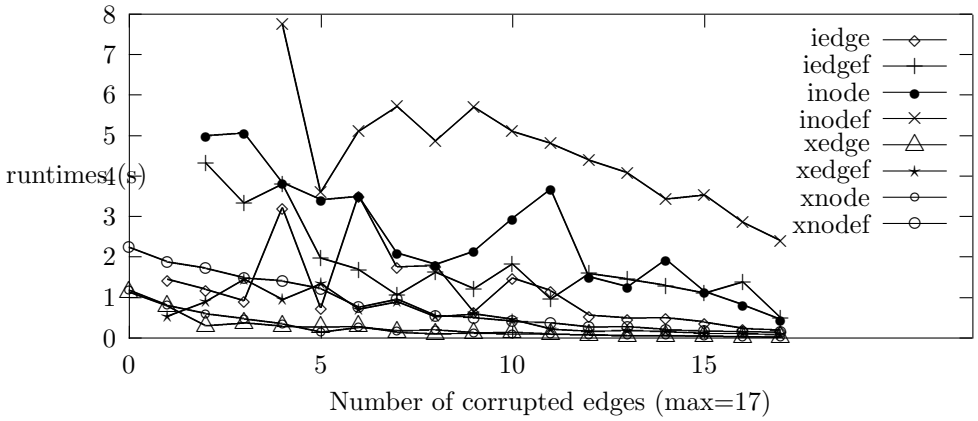

Fig. 6. Runtimes for fisheries model

6 Discussion

In extending QCM to time-based simulations, TQCM$^{\texttt{xnode}}$ was best, with TQCM$^{\texttt{iedge}}$ a close second. Since TQCM$^{\texttt{inode}}$ performed so poorly, this study is pessimistic about the practicality of the Menzies & Cohen 3-copy-limit optimisation. However, in a personal communication, Cohen reports early results indicating that a small variant on the 3-copy-limit proof for TQCM$^{\texttt{inode}}$ may prove that there is only a 2-copy-limit for TQCM$^{\texttt{iedge}}$. That is, if we are willing to have a small inexplicable rate for correct models (5%), we can run TQCM$^{\texttt{iedge}}$ models for very long periods of time. This is an exciting possibility which is currently being actively explored.

Our approach is practical in domains where practioners can build an automatic test engine: a non-trivial task. Our original work took weeks to generate a single line on Fig. 5. However, once the test engine was operational, the utility of modelling options can be assessed very quickly. For example, whilst writing this article, we detected a small data collection error. Re-running all the linking options took less than three days (the results shown here come from the re-run).

One drawback with the current study is that its conclusions are very dependent on our experiments with a single model (fisheries). Note that this restriction does not invalidate the evaluation method proposed here. Further, our current mutator did not vary the topology of the graph, merely its edge annotations. The next generation of mutators will address this issue.

7 Conclusion

We have offered a general framework for evaluating a QRS. We propose critical success metrics (e.g. Fig. 1) to test the utility of modelling options within a QRS. Three general QRS CSMs are *accuracy, restrictiveness* and *practicality* (§2). A useful tool for this process is a model mutator (§4) that can build test models from representative models. A desirable property of such a mutator is *graded degradation*, e.g Fig. 1; i.e. a method of generating a range of `good` to `poor` models.

This framework has been used to test linking policies for qualitative temporal models, and subsequently identifying three useful methods.

This work represents an incremental advance towards our long term research objective of establishing methodologies for OR model validation. Immediate future work includes decreasing the number of measurements made, increasing the number of time steps modelled, and studies with other simulation models.

Acknowledgment

Kingsley Jones (DSTO) initially proposed the model mutator used in this study.

References

1. H. Bossel. *Modeling and Simulations*. A.K. Peters Ltd, 1994. ISBN 1-56881-033-4.
2. I. Bratko, I. Mozetic, and N. Lavrac. *KARDIO: a Study in Deep and Qualitative Knowledge for Expert Systems*. MIT Press, 1989.
3. T. Bylander. A Critique of Qualitative Simulation from a Consolidation Viewpoint. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(2):252–263, March/April 1988.
4. B. Feldman, P. Compton, and G. Smythe. Towards Hypothesis Testing: JUSTIN, Prototype System Using Justification in Context. In *Proceedings of the Joint Australian Conference on Artificial Intelligence, AI '89*, pages 319–331, 1989.
5. Y. Iwasaki. Qualitative Physics. In P.R. Cohen A. Barr and E.A. Feigenbaum, editors, *The Handbook of Artificial Intelligence*, volume 4, pages 323–413. Addison Wesley, 1989.
6. B. Kuipers. Qualitative Simulation. *Artificial Intelligence*, 29:229–338, 1986.

7. R. Levins and C.J. Puccia. *Qualitative Modeling of Complex Systems: An Introduction to Loop Analysis and Time Averaging*. Harvard University Press, Cambridge, Mass., 1985.
8. T.J. Menzies and R.E. Cohen. "And" Can You Validate It? In *Submitted to the Australian AI '97 conference.*, 1997.
9. T.J. Menzies and R.E. Cohen. A Graph-Theoretic Optimisation of Temporal Abductive Validation. In *European Symposium on the Validation and Verification of Knowledge Based Systems, Leuven, Belgium*, 1997.
10. T.J. Menzies and P. Compton. Knowledge Acquisition for Performance Systems; or: When can "tests" replace "tasks"? In *Proceedings of the 8th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada*, 1994.
11. T.J. Menzies and P. Compton. Applications of Abduction: Hypothesis Testing of Neuroendocrinological Qualitative Compartmental Models. *Artificial Intelligence in Medicine*, 1997. To appear.
12. K.M. Yip. Understanding Complex Dynamics by Visual and Symbolic Reasoning. *Artificial Intelligence*, 51:179–221, 1991.